J Cryptogr Eng (2016) 6:75–82
DOI 10.1007/s13389-015-0114-1

REGULAR PAPER

The BRUTUS automatic cryptanalytic framework

Testing CAESAR authenticated encryption candidates for weaknesses

Markku-Juhani O. Saarinen[1]

Received: 16 January 2015 / Accepted: 15 November 2015 / Published online: 7 December 2015

Abstract This report summarizes our results from security analysis covering all 57 competitions for authenticated encryption: security, applicability, and robustness (CAESAR) first-round candidates and over 210 implementations. We have manually identified security issues with three candidates, two of which are more serious, and these ciphers have been withdrawn from the competition. We have developed a testing framework, BRUTUS, to facilitate automatic detection of simple security lapses and susceptible statistical structures across all ciphers. From this testing, we have security usage notes on four submissions and statistical notes on a further four. We highlight that some of the CAESAR algorithms pose an elevated risk if employed in real-life protocols due to a class of adaptive-chosen-plaintext attacks. Although authenticated encryption with associated data are often defined (and are best used) as discrete primitives that authenticate and transmit only complete messages, in practice, these algorithms are easily implemented in a fashion that outputs observable ciphertext data when the algorithm has not received all of the (attacker-controlled) plaintext. For an implementor, this strategy appears to offer seemingly harmless and compliant storage and latency advantages. If the algorithm uses the same state for secret keying information, encryption, and integrity protection, and the internal mixing permutation is not cryptographically strong, an attacker can exploit the ciphertext–plaintext feedback loop to reveal secret state information or even keying material. We conclude that the main advantages of exhaustive, automated cryptanalysis are that it acts as a very necessary sanity check for implementations and gives the cryptanalyst insights that can be used to focus more specific attack methods on given candidates.

Keywords Authenticated encryption · CAESAR · BRUTUS · Adaptive-chosen-plaintext attacks · Automated cryptanalysis

1 Introduction

Authenticated encryption with associated data (AEAD) algorithms provide message confidentiality and integrity protection with a single cryptographic primitive. As such, they offer functionality similar to combining a stream or block cipher with a message authentication code (MAC) on protocol level.

This two-algorithm approach has been the predominant way of securing messages in popular Internet security protocols since mid-1990's. Its potential problems were identified early by Krawczyk and others [22]. Still, current TLS 1.2 [11] mandates support only for the `TLS_RSA_WITH_AES_128_CBC_SHA` cipher suite, which combines AES [28] in CBC [12] Confidentiality Mode with SHA-1 [31] hash algorithm in HMAC [30] Message Authentication mode, and a TLS-specific padding scheme. Similar approaches have been taken by other popular security protocols such as IPSec [20,21] and SSH [52]. This separation has been exploited by numerous real-life attacks [10,33,45].

When authenticated encryption techniques such as GCM [29] are used, most problems related to intermixing of two separate algorithms (such as padding) are removed. Furthermore, AES-GCM works in a single pass, resulting in

Much of this research was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme, hosted at Department of Telematics, NTNU, Norway.

✉ Markku-Juhani O. Saarinen
m.saarinen@qub.ac.uk; mjos@iki.fi

[1] ECIT, Queen's University of Belfast, Northern Ireland Science Park, Queen's Road, Queen's Island, Belfast BT3 9DT, UK

increased throughput and a decreased implementation footprint. AES-GCM has rapidly replaced older methods in practical usage. It is endorsed and effectively enforced for US and Allied National Security Systems [9]. AES-GCM has been adopted for use in many protocols, including TLS, SSH, and IPSec [7,18,41]. However, GCM is widely seen as an unsatisfactory standard with brittle security assurances [32] and therefore a new NIST-sponsored competition, competition for authenticated encryption: security, applicability, and robustness (CAESAR), was launched in 2014 [8]. The CAESAR competition has multiple stages or "elimination rounds." The call for algorithms resulted in 57 first-round candidates.

Structure of this paper and our contributions We give a description of AEAD that most CAESAR candidates conform to in Sect. 2. We started our evaluation by getting to know the voluminous supplied documentation, which led to cryptanalytic results on three candidates (Sect. 3). We then describe the development of our framework for automated cryptanalysis, BRUTUS, in Sect. 4, together with security usage notes obtained. A key observation which may not have been fully considered by all submitters is the non-atomic nature of AEAD in real life, which is captured in the notion of adaptive-chosen-plaintext attacks (Sect. 5). The candidates can be classified according to their robustness against adaptive-chosen-plaintext attacks, which generally do not apply to AES-GCM. This is done in Sect. 6, and we conclude in Sect. 7.

2 Authenticated encryption with associated data

Most CAESAR authenticated encryption algorithms with associated data (AEAD) algorithms have the following inputs:

- K A secret, shared confidentiality, and integrity key.
- N Public nonce or initialization vector. Optionally transmitted.
- P Message payload, for which both confidentiality and integrity is protected.
- A Associated "header" data. These data are only authenticated.[1]

The AEAD transform will output a single binary string C that contains additional entropy bits for detection of modifications:

$$\mathsf{AEAD}(K, N, P, A) \rightarrow C. \tag{1}$$

[1] Associated data A may be transmitted unencrypted or implicitly known to both parties (meta information such as message sequence numbers and endpoint identities).

The inverse transform will only return the original message payload P if correct values for K, N, A, and C are supplied:

$$\mathsf{AEAD}^{-1}(K, N, C, A) \rightarrow P \text{ or } \mathsf{FAIL}. \tag{2}$$

We may semi-formally characterize the security requirements for AEAD and AEAD^{-1} which are relevant to this work as follows:

1. *Confidentiality* Even if a large number of chosen (N, P, A) (with non-repeating N) can be supplied by an attacking algorithm to an encryption oracle $\mathsf{AEAD}(?, N, P, A) \rightarrow C$, it should be infeasible to distinguish the corresponding outputs C from equal-length random string.
2. *Integrity* It should be infeasible to create any new set (N, C, A) that would *not* output $\mathsf{AEAD}^{-1}(?, N, C, A) = \mathsf{FAIL}$ for an unknown key, even if a very large number of valid (N, P, C, A) sets for that secret key are available.

More trivial security properties follow from these requirements. Each submission was free to define what "infeasible" in their particular case means. For the confidentiality requirement, this is traditionally expected to mean effort commensurate with an exhaustive search for the secret key K. The forgery effort (integrity goal) depends on the size of authentication variable (message expansion from P to C), but can be defined to be lower. For example, AES-GCM archives a significantly lower level of integrity protection than information theoretically expected [34,37]. As CAESAR is a cryptographic *competition*, we may consider all such suboptimal features to be relative weaknesses.

3 Manual cryptanalysis

CAESAR candidates came in many shapes and sizes. We refer to [1] and the authenticated encryption zoo web site for classification and current status of each one of the candidates.[2] Here's our rough breakdown:

- 8 Clearly based on the Sponge construction.
- 9 Somehow constructed from AES components.
- 19 AES modes of operation.
- 21 Based on other design paradigms or just ad hoc.

A group of proposals cannot be even evaluated according to established cryptologic criteria and we sidestep those in this report.

We spent some time familiarizing ourselves with the substantial amount of technical documentation after it was released in March 2014. Based on the specifications alone, we identified clear cryptanalytic problems with three candidates:

[2] https://aezoo.compute.dtu.dk/.

1. *PAES* [51] suffered from rotational cryptanalytic flaws as round constants were not used. Similar observations were made simultaneously by Sasaki and Wang [42] and Jean and Nikolić [19] teams. PAES has been withdrawn from the CAESAR competition.
2. *HKC* [16] was found to suffer from an almost linear authentication function, which could be used for high-probability message forgeries. HKC has been withdrawn from the CAESAR competition.
3. *iFeed[AES]* [53]. We offered criticism towards this proposal as the authentication tag depends only on the last block of the plaintext.[3]

4 Exhaustive methodology: the BRUTUS framework

By June 2014, most of the 57 teams had submitted reference implementations for their candidates. Many of these candidates had multiple parameter choices and optimizations, bringing the total number of implementations to over 210.

The implementations were integrated into the SUPERCOP[4] speed testing framework by D. Bernstein. In addition to very rudimentary coherence testing, the sole functionality of SUPERCOP is in performance measurement. SUPERCOP is not very well suited for statistical testing or other experimental work.

4.1 Development process

We decided to build our own testing framework which would allow more rapid experimentation. We lifted the reference implementations from the SUPERCOP framework as we had no use for it. Our BRUTUS[5] toolkit compiles each reference implementation into a dynamically linked library that can be loaded "on the fly" into an arbitrary experimentation program. The standard test module performs coherence testinga and speed tests, and generates test vectors known as known answer tests (KATs). Interfacing with arbitrary languages can be archived via small native components.

Due to the disappointingly poor quality of some of the code (even from some prominent cryptologic security teams), many implementations had to be corrected to fix memory leaks and other elementary errors that affected stability of experimentation. We avoided modifying the mathematical structure of the implementations even when it appeared to contradict the supplied documentation. BRUTUS is intended purely as a research and experimentation tool.

[3] Similar issues apply to some other proposals such as OCB [24] and OTR [25], which restricts their usage in protocols where some level of collision resistance is expected.

[4] http://bench.cr.yp.to/supercop.html.

[5] https://github.com/mjosaarinen/brutus/.

4.2 Identifying ciphers and modes

An interesting advantage gained from having a coherent and easy interface for all ciphers is that an "identifying gallery"[6] of proposed modes and ciphers can be constructed. This allows black-box identification of ciphers in some cases. The diagrams are independent of secret keying information. Figure 1 shows some members of this gallery.

4.3 Implementability and side channels

It is clear that some proposals are poorly suited for hardware-only implementation. For example, any algorithm actually requiring `malloc()` dynamic memory allocation—which in itself is a side channel security headache—is difficult to implement in hardware. How this will be addressed is left to the CAESAR committee as hardware implementations are not expected before the second round. Some proposals have been implemented in FPGA. The proposed SÆHI API allows generic, hybrid software-hardware implementations and is therefore able to cover almost all candidates [40]. BRUTUS is capable of supporting this API.

4.4 Performance

We refer to SUPERCOP results for software performance metrics across a number of implementation targets. Speed-optimized implementations were not even expected for first-round candidates, so such comparisons would be unfair (the call was for "readable" implementations, which was rather liberally interpreted by some teams). Efficient implementation of parallelized modes in plain ANSI C is nontrivial. As a generic note, none of the proposed AES modes seem to reach the *authentication* speeds attained by AES-GCM—thanks to AES-NI finite field instructions that directly support GCM. Furthermore, some modes are not entirely parallel, and therefore cannot reach the maximum throughput speeds attainable by AES-GCM and offer little or no advantage over it. We urge careful analysis of these factors during selection.

4.5 Security usage notes on various ciphers

We tested basic forgery strategies, the effect of key and nonce modifications to ciphertext, and diffusion of changes in the cipher state. From our automated testing, we arrived at the following notes:

1. *CMCC* [44] does not use all of its keying materials for short messages, and therefore, a trivial forgery can be made even if a part of the secret key is not known. The author has proposed a tweak.

[6] https://mjos.fi/aead_feedback/.

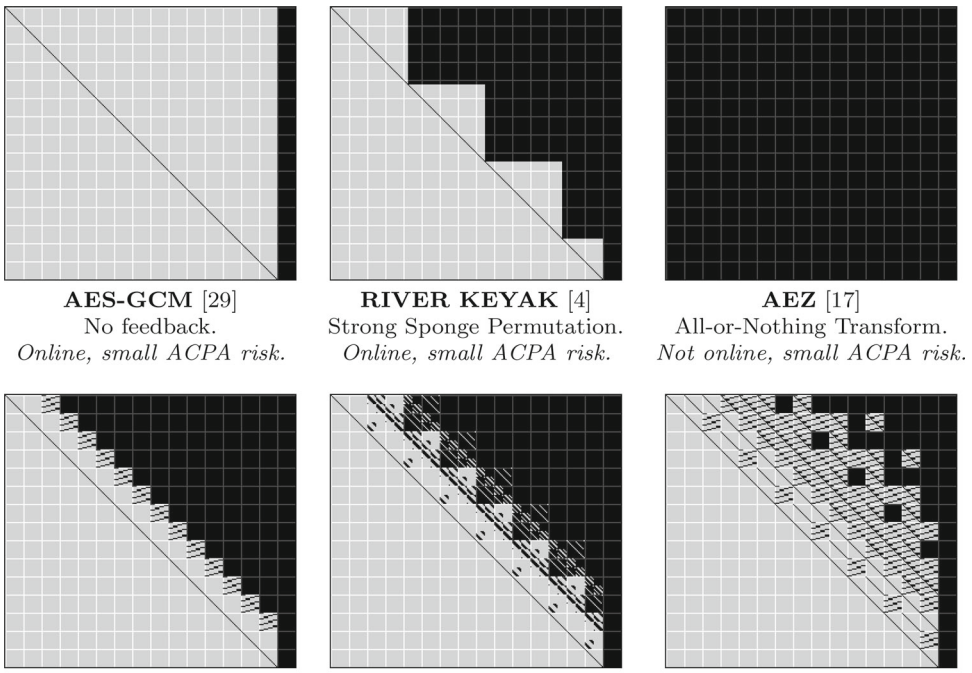

Fig. 1 Visualization of feedback properties of some CAESAR candidates. Here each pixel represents a single byte. *Grid lines* are every 16 bytes (128 bits). The Y coordinate is the single plaintext byte change location offset. Each *pixel line* represents 256 bytes of ciphertext difference, with affected ciphertext bytes *darkened*. The authentication tag is usually seen as a *bar on the right side*; those bytes are affected by any change. The "ripples" on the *lower three diagrams* are one indication of inconsistent mixing

2. *CALICO* [43] had an extraordinarily long key ($32+16=48$ bytes), which consists of a 32-byte decryption key and a 16-byte MAC key. If you have a false key (with something else in the first key 32 bytes), CALICO will not detect it and will just output nonsense. This can be circumvented in implementations but does violate basic AEAD security expectations. The author withdrew CALICO from the competition earlier.
3. *PAEQ* [5] implementations exhibited a property in which authentication of associated data only (i.e., no payload) did not depend on the supplied nonce at all, leading to replay forgery attacks in case a protocol is sending A only. The authors noted that the specification forbids such messages (but were allowed in actual implementation for compatibility), but are working on a tweak. We encourage such a tweak as this would make the proposal plug-in compatible with AES-GCM in security protocols where signaling frequently demands authentication of metadata only.
4. *YEASv2* [6]. Although it is mentioned the specification, the nonce has only 127 effective bits. The ignored bit is bit 0 of the last of byte of the 16-byte IV sequence. This is an unfortunate selection; if we are using network (big endian) byte order, this is the least significant bit of the nonce. If running sequence numbers are used, every two consecutive messages will have equivalent nonces and security will break.

All of these issues are fairly easy to address. Again we ignore less professional proposals that do not meet basic sanity and CAESAR compliance criteria.

4.6 Implementation security

Based on our cursory code review of the 210+ implementations, our general advice is strongly against using CAESAR reference ciphers as a part of any real-life application requiring stability or security at this stage of competition.

5 Most AEAD are not atomic

When described in the fashion of Eqs. 1 and 2, an AEAD transform appears to be an atomic, indivisible operation. Two-pass CAESAR candidates can essentially only be implemented this way. The AEZ [17] and SIV [23] candidates are examples of such "All-or-nothing Transforms" [35].

Due to efficiency and memory conservation reasons, most CAESAR candidates can work in "online" mode where the full plaintext block P is not required for the encryption algorithm to be able to produce some of the ciphertext. This is generally done by dividing the message to uniform-sized message blocks $\mathsf{pad}(M) = M_1 \,||\, M_2 \,||\, \cdots \,||\, M_n$. The AEAD maintain an internal state X which is initialized with some value derived from K and N. This is then iterated over

blocks M_i and the final state is subjected to another transformation to produce a MAC tag T.

$X_0 = \mathsf{key}(K, N)$	Initialize state from key and nonce.
$X_i = \mathsf{mix}(M_i, X_{i-1})$	Mix message blocks with state, $1 \leq i \leq n$.
$C_i = \mathsf{out}(X_i)$	Block derived from state, $1 \leq i \leq n$.
$T = \mathsf{fin}(X_n)$	Finalization—compute the authentication tag.

The ciphertext is constructed as

$$C = C_1 \,||\, C_2 \,||\, \cdots \,||\, C_n \,||\, T. \tag{3}$$

This type of construction allows C_i to be output immediately after M_i is fed into the mixing transform. All Sponge-based [3] constructions and many proposed block cipher modes of operation fall into this category.

5.1 The adaptive-chosen-plaintext attack

The adaptive-chosen-plaintext attack applies to AEAD designs which are *not* necessarily based on block ciphers at all. We assume that an attacker can adaptively feed a plaintext block M_i to the cipher as a function of previously observed ciphertext blocks

$$M_i = f_{\mathsf{atk}}(C_1, C_2, \ldots, C_{i-1}). \tag{4}$$

The attacker function f_{atk} can perform some reasonable amount of computation for the feedback operation.

We argue that this is a relevant model offering insights especially to smart card applications and other lightweight applications where an attacker has full control over the communication channel.

The goal of the attacker is to derive information about the internal state X_i. This information can be used in attacks of various degrees of severity:

1. Distinguish or partially predict C_{i+1}.
2. Fully derive X_i; predict all future C_i and T.
3. Derive information about K.

Note that message authentication is not an issue in an adaptive-chosen-plaintext attack on an AEAD as encryption cannot really fail. The inverse scenario of Eq. 4, a chosen ciphertext attack, is less realistic as it would seem to automatically break the definition given by Eq. 2. However, this scenario has been considered in the literature [2].

6 CAESAR candidates and real-life protocols: susceptibility to adaptive-chosen-plaintext attacks

In order to integrate a CAESAR AEAD into a real-life protocol such as TLS, SSH, or IPSec, one has to define not only the appropriate ciphersuite identifiers but also the usage and formatting mechanisms.

In case of all AEAD, an obvious path of integration is to adopt the mechanisms used for AES-GCM in relevant RFCs: TLS in [41], SSH in [18], and IPSec in [7]. This will allow implementors to essentially "plug in" the algorithms into existing protocol implementation frameworks. In many protocol instances, the ciphers are subjected to adaptive-chosen-plaintext attacks with relative ease.

Even though the CAESAR call for algorithms[7] was careful to require concrete security claims for full AEAD transforms, the security claims related to this type of attack are not explicitly stated for many ciphers. However, internal mixing qualities of a design offers a direct insight into the robustness of a cipher against adaptive-chosen-plaintext attacks.

Based on our automated analysis, at least ACORN [47], AEGIS [50], MORUS [48], and TIAOXIN [27] represent significantly elevated adaptive-chosen-plaintext attack risk. We are formalizing our observations, but we note that—as an example—the effective internal state can be trivially forced to be smaller, helping birthday attacks. These proposals have a single state without separation between authentication, confidentiality, or keying state. In this, they are similar to Sponge designs. Indeed, if these had been *labeled* "sponge designs," they could be declared "broken" due to the weakness of their mixing functions. This illustrates the difficulty of security comparisons among candidates.

In many ways, these ciphers resemble Helix [15] and Phelix [46], which were proposed as an authenticated stream ciphers a decade ago. These ciphers were attacked in under various assumptions [26,49]. Another earlier, similar (but lightweight) authenticated design is the Hummingbird cipher [13,14], which was successfully cryptanalyzed [36,38].

These ciphers seem to have been created with ad hoc design methods and offer no provable security assurances. This by no means indicates that they cannot be used securely and use of these candidates may be highly justified in many cases as they are among fastest (or, in case of ACORN, smallest) candidates.

In comparison, we offer the following proof sketches for resistance of certain other essential classes of algorithms to adaptive-chosen-plaintext attacks of this type.

Theorem 1 *AES-GCM is not vulnerable to adaptive-chosen-plaintext attacks.*

[7] http://competitions.cr.yp.to/caesar-call.html.

Proof The Galois/counter mode has an essentially independent counter mode and a polynomial-based authentication mechanism. Since the counter mode keystream can be generated *a priori* to encryption, any ciphertext–plaintext feedback will not yield useful information about the internal state of the mode. □

Theorem 2 *Sponge modes with strong permutations such as* DUPLEXWRAP [4] *or* BLNK [39] *are not vulnerable to adaptive-chosen-plaintext attacks.*

Proof These modes utilize a cryptographically strong permutation between any two blocks of data, and therefore, the adaptive attacker has no access to capacity beyond that barrier. □

As there are some proposals that employ various stronger notions of provable security, we make the following general observation:

Observation 1 *Provably secure modes that have two or more passes over data are not vulnerable to adaptive-chosen-plaintext attacks.*

Figure 1 offers a visualization of Theorems 1 and 2 and the final observation.

7 Conclusions and further work

We have presented a summary of our initial examination and analysis covering all 57 CAESAR first-round proposals (we are only presenting results that we have obtained ourselves). As an executive note, we strongly recommend against using any of the first-round CAESAR ciphers in real-life applications despite their novelty and often famous authorship.

During manual examination, we have identified cryptographic problems with three proposals, two of which have been withdrawn from the competition.

We have described our development of the BRUTUS testing framework which allows tests to be made that automatically cover all candidates. As performance testing was not even required in the first round (and is adequately addressed by the SUPERCOP toolkit), we focused on the structural differences of various candidates. We offer security usage notes for four candidates.

From the BRUTUS automated tests, we observe that some candidates offer less than convincing resistance against adaptive-chosen-plaintext attacks. This is significant since one of the main motivations for the CAESAR competition is to seek secure replacements for the AES-GCM algorithm which is provably secure against this type of attack. Sponge permutation designs and two-pass provably secure modes are also resistant. Such an attack can be mounted with relative ease in conceivable instances of real-life protocols such as TLS, SSH, and IPSec.

Based on our experience, the most valuable output from exhaustive, automated testing across actual cipher implementations is that it catches implementation errors and possible errors in *security usage*—discrepancies between the assumptions of the users of the algorithm and its designers. These often break real-life protocols and applications that utilize encryption algorithms. The insights obtained from statistical testing of (internal) quantities can be used by a cryptanalyst to focus more specific analysis efforts against those candidates that are expected to be vulnerable to a particular method of attack.

We intend to extend this work to performance analysis, analysis of hardware implementations, and statistical analysis of the internal cipher state for the second-round CAESAR candidates.

References

1. Abed, F., Forler, C., Lucks, S.: General overview of the first-round CAESAR candidates for authenticated encryption. IACR ePrint 2014/792 (2014). https://eprint.iacr.org/2014/792. Accessed 27 Nov 2015
2. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. IACR ePrint 2014/144 (2014). https://eprint.iacr.org/2014/144. Accessed 27 Nov 2015
3. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the sponge: single-pass authenticated encryption and other applications. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. *LNCS*, vol. 7118, pp. 320–337. Springer, New York (2011). doi:10.1007/978-3-642-28496-0_19
4. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V., Keer, R.V.: CAESAR submission: Keyak v1. CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/keyakv1.pdf. Accessed 27 Nov 2015
5. Biryukov, A., Khovratovich, D.: PAEQv1. CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/paeqv1.pdf. Accessed 27 Nov 2015
6. Bosselaers, A., Vercauteren, F.: YAES v2. CAESAR first round candidate (2014). http://competitions.cr.yp.to/round1/yaesv2.pdf. Accessed 27 Nov 2015
7. Burgin, K., Peck, M.: Suite B profile for internet protocol security (IPsec). IETF RFC 6380 (2011)
8. CAESAR: CAESAR: competition for authenticated encryption: security, applicability, and robustness (2014). http://competitions.cr.yp.to/caesar.html. Accessed 27 Nov 2015
9. CNSSP: National information assurance policy on the use of public standards for the secure sharing of information among national security systems. CNSS Policy No. 15 (2012)
10. Degabriele, J.P., Paterson, K.G.: On the (in)security of IPsec in MAC-then-encrypt configurations. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 493–504. ACM (2010)

11. Dierks, T., Rescorla, E.: The transport layer security (TLS) protocol version 1.2. IETF RFC 5246 (2008). https://tools.ietf.org/html/rfc5246. Accessed 27 Nov 2015
12. Dworkin, M.: Recommendation for block cipher modes of operation. NIST Special Publication 800-38A (2001)
13. Engels, D., Fan, X., Gong, G., Hu, H., Smith, E.M.: Hummingbird: ultra-lightweight cryptography for resource-constrained devices. In: Sion, R., Curtmola, R., Dietrich, S., Kiayias, A., Miret, J.M., Sako, K., Sebé, F. (eds.) Financial Cryptography and Data Security, FC 2010 Workshops. LNCS, vol. 6054, pp. 3–18. Springer, New York (2010)
14. Engels, D., Saarinen, M.J.O., Schweitzer, P., Smith, E.M.: The Hummingbird-2 lightweight authenticated encryption algorithm. In: Juels, A., Paar, C. (eds.) RFIDSec'11. LNCS, vol. 7055, pp. 19–31. Springer, New York (2011)
15. Ferguson, N., Whiting, D., Schneier, B., Kelsey, J., Lucks, S., Kohno, T.: Helix: fast encryption and authentication in a single cryptographic primitive. In: Johansson, T. (ed.) FSE'03. LNCS, vol. 2887, pp. 330–346. Springer, New York (2003)
16. Henricksen, M., Kiyomoto, S., Lu, J.: The HKC authenticated stream cipher (ver. 1). CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/hkcv1.pdf. Accessed 27 Nov 2015
17. Hoang, V.T., Krovetz, T., Rogaway, P.: AEZ v1: authenticated-encryption by enciphering. CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/aezv1.pdf. Accessed 27 Nov 2015
18. Igoe, K.: Suite B cryptographic suites for secure shell (SSH). IETF RFC 6239 (2011). https://tools.ietf.org/html/rfc6239. Accessed 27 Nov 2015
19. Jean, J., Nikolić, I.: Using AES round symmetries to distinguish PAES (2014). http://www1.spms.ntu.edu.sg/~syllab/m/images/6/6e/Using_AES_Round_Symmetries_to_Distinguish_PAES.pdf. Accessed 27 Nov 2015
20. Kent, S.: IP authentication header. IETF RFC 4302 (2005). https://tools.ietf.org/html/rfc4302. Accessed 27 Nov 2015
21. Kent, S.: IP encapsulating security payload (ESP). IETF RFC 4303 (2005). https://tools.ietf.org/html/rfc4303. Accessed 27 Nov 2015
22. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: how secure is SSL?). In: Kilian, J. (ed.) CRYPTO'01. LNCS, vol. 2139, pp. 310–331. Springer, New York (2001)
23. Krovetz, T.: HS1-SIV (v1). CAESAR 1st round candidate (2014). http://competitions.cr.yp.to/round1/hs1sivv1.pdf. Accessed 27 Nov 2015
24. Krovetz, T., Rogaway, P.: OCB (v1). CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/ocbv1.pdf. Accessed 27 Nov 2015
25. Minematsu, K.: AES-OTR v1. CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/aesotrv1.pdf. Accessed 27 Nov 2015
26. Muller, F.: Differential attacks against the helix stream cipher. In: Roy, B., Meier, W. (eds.) FSE'04. LNCS, vol. 3017, pp. 94–108. Springer, New York (2004)
27. Nikolić, I.: Tiaoxin-346, version 1.0. CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/tiaoxinv1.pdf. Accessed 27 Nov 2015
28. NIST: Advanced encryption standard (AES). Federal Information Processing Standards Publication FIPS 197 (2001). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Accessed 27 Nov 2015
29. NIST: Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. NIST Special Publication 800-38D (2007). http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf. Accessed 27 Nov 2015
30. NIST: The keyed-hash message authentication code (HMAC). Federal Information Processing Standards Publication FIPS 198-1 (2008)
31. NIST: Secure hash standard (SHS). Federal Information Processing Standards Publication FIPS 180-4 (2012). http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf. Accessed 27 Nov 2015
32. NIST VCAT: NIST cryptographic standards and guidelines development process: report and recommendations of the visiting committee on advanced technology of the national institute of standards and technology (2014)
33. Paterson, K.G., Yau, A.K.L.: Cryptography in theory and practice: the case of encryption in IPsec. In: Vaudenay, S. (ed.) EUROCRYPT'06. LNCS, vol. 4004, pp. 12–29. Springer, New York (2006)
34. Procter, G., Cid, C.: On weak keys and forgery attacks against polynomial-based MAC schemes. In: Moriai, S. (ed.) FSE'13. LNCS, vol. 8424, pp. 287–304. Springer, New York (2013)
35. Rivest, R.: All-or-nothing encryption and the package transform. In: Biham, E. (ed.) FSE'97. LNCS, vol. 1267, pp. 210–218. Springer, New York (1997)
36. Saarinen, M.J.O.: Cryptanalysis of Hummingbird-1. In: Joux, A. (ed.) FSE'11. LNCS, vol. 6733, pp. 328–341. Springer, New York (2011)
37. Saarinen, M.J.O.: Cycling attacks on GCM, GHASH and other polynomial MACs and hashes. In: Canteaut, A. (ed.) FSE'12. LNCS, vol. 7549, pp. 216–225. Springer, New York (2012). doi:10.1007/978-3-642-34047-5_13
38. Saarinen, M.J.O.: Related-key attacks against full Hummingbird-2. In: Moriai, S. (ed.) FSE'13. LNCS, vol. 8424, pp. 467–482. Springer, New York (2013)
39. Saarinen, M.J.O.: Simple AEAD hardware interface (SÆHI) in a SoC: implementing an on-chip Keyak/WhirlBob coprocessor. In: TrustED'14 Proceedings of the 4th International Workshop on Trustworthy Embedded Device, pp. 51–56. ACM (2014). doi:10.1145/2666141.2666144
40. Saarinen, M.J.O.: The STRIBOBr1 authenticated encryption algorithm. CAESAR, 1st round candidate (2014). http://www.stribob.com. Accessed 27 Nov 2015
41. Salter, M., Housley, R.: Suite B profile for transport layer security (TLS). IETF RFC 6460 (2012). https://tools.ietf.org/html/rfc6460. Accessed 27 Nov 2015
42. Sasaki, Y., Wang, L.: A practical universal forgery attack against paes-8. IACR ePrint 2014/218 (2014). https://eprint.iacr.org/2014/218. Accessed 27 Nov 2015
43. Taylor, C.: The Calico family of authenticated ciphers, version 8. CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/calicov8.pdf. Accessed 27 Nov 2015
44. Trostle, J.: AES-CMCC v1.1. CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/aescmccv11.pdf. Accessed 27 Nov 2015
45. Vaudenay, S.: Security flaws induced by CBC padding—applications to SSL, IPSEC, WTLS. In: Knudsen, L.R. (ed.) EUROCRYPT'02. LNCS, vol. 2332, pp. 534–546. Springer, New York (2002)
46. Whiting, D., Schneier, B., Lucks, S., Muller, F.: Phelix—fast encryption and authentication in a single cryptographic primitive. ECRYPT Stream Cipher Project Report 2005/027 (2005). http://www.schneier.com/paper-phelix.html. Accessed 27 Nov 2015
47. Wu, H.: ACORN: a lightweight authenticated cipher (v1). CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/acornv1.pdf. Accessed 27 Nov 2015
48. Wu, H., Huang, T.: The authenticated cipher MORUS (v1). CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/morusv1.pdf. Accessed 27 Nov 2015

49. Wu, H., Preneel, B.: Differential-linear attacks against the stream cipher phelix. eSTREAM preprint (2006). http://www.ecrypt.eu.org/stream/papersdir/2006/056.pdf. Accessed 27 Nov 2015
50. Wu, H., Preneel, B.: AEGIS: a fast authenticated encryption algorithm (v1). CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/aegisv1.pdf. Accessed 27 Nov 2015
51. Ye, D., Wang, P., Hu, L., Wang, L., Xie, Y., Sun, S., Wang, P.: PAES v1: parallelizable authenticated encryption schemes based on AES round function. CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/paesv1.pdf. Accessed 27 Nov 2015
52. Ylönen, T., Lonvick, C.: The secure shell (SSH) connection protocol. IETF RFC 4254 (2006). https://tools.ietf.org/html/rfc4254. Accessed 27 Nov 2015
53. Zhang, L., Wu, W., Sui, H., Wang, P.: iFeed[AES] v1. CAESAR first round submission (2014). http://competitions.cr.yp.to/round1/ifeedaesv1.pdf. Accessed 27 Nov 2015